Homework 11: Arrays

CS 1323/4 Fall 2023

Name: Arboy Magomba

Student ID (usually 112-XXX-XXXX or 113-XXX-XXXX): 113-650-204

1. (5 points) Trace the code below using the memory diagram. Remember to trace index. Remember to initialize the array contents when an array is constructed.

```
int[] data = new int[6];
int[] copy = {1, 7, 5, 3};
for (int index = 0; index < copy.length; ++ index) // read carefully
{
        data[index] = copy[index];
        data[index + 2] = copy[index];
}
```

main stack frame		
Identifier	**Address**	**Contents**
data	100	1000
copy	101	1007
	102	
	103	
	104	

Heap		
Identifier	**Address**	**Contents**
0	1000	0, 1
1	1001	0, 7
2	1002	0, 1, 5
3	1003	0, 7, 3
4	1004	0, 5
5	1005	0, 3
length	1006	6
0	1007	1
1	1008	7
2	1009	5
3	1010	3
length	1011	4
	1012	
	1013	
	1014	
	1015	
	1016	
	1017	
	1018	

2. (9 points; 3 points each) Trace each loop in the table below.

a.

```
double[] data = {9.1, 8.4, 3.2, 7.2, 6.3, 5.6};
double temp = data[0];
data[0] = data[5];
data[5] = temp;
temp = data[1];
data[1] = data[4];
data[4] = temp;
temp = data[2];
data[2] = data[3];
data[3] = temp;
```

temp	data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]
9.1	9.1	8.4	3.2	7.2	6.3	5.6	
8.4	5.6	6.3	7.2	3.2	8.4	9.1	
3.2							

b.

```
String[] arr = {"Whose", "woods", "these", "are", "I", "think", "I", "know"};
for (int index=0; index<arr.length; ++index)
{
	String element = arr[index];
	char first = element.charAt(0);
	char last = element.charAt(element.length() - 1);
	if (first > last) // Remember all capital letters are before
	// all small ones in ASCII
		arr[index] = "Frost";
}
```

index	element	first	last	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]	arr[8]
0	Whose	W	e	Whose, Frost	woods	these	are	I	think	I	know	
1	woods	w	s		woods							
2	these	t	e			these						
3	are	a	e				Frost					
4	I	I	I					I				
5	think	t	k						think			
6	I	I	I							I		
7	know	k	w								Frost	

c.

```
int[] data = {4, 2, 5, 8, 6, 3, 7, 1};
for (int index = 0; index < data.length - 1; ++index) // read carefully
{
	if (data[index] > data[index + 1])
	{
		data[index + 1] = data[index];
	}
}
```

index	data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]	data[7]
	4	2	5	8	6	3	7	1
0		4						
1								
2								
3					8			
4						6		
5								
6								7

3. (6 points, 3 points each) Use **specified method(s) in Arrays class** to perform the operations below in a code fragment. You must use the specified methods in the Arrays class to get credit.

   a. Write a method that will return a new array that contains the elements that occur after the second occurrence of the first element of the array. If the first element of the array does not occur later in the array, a zero length array should be returned. Zero length arrays are legal in Java. Use the copyOfRange method.

   For example: if array passed to source contained {8, 4, 7, 2, 5, 7, 9, 8, 1, 7}, then the returned array should contain {1, 7}. If the array passed to source contained {1, 2, 3, 4, 5}, then a zero length array {} should be returned.

   Hint: Think carefully about which direction it would be best to search in.

   The method signature is below:

   public static int[] afterSecondOccurrenceOfFirstElement(int[] source)

```
public static int[] afterSecondOccurrenceOfFirstElement(int[] input) {
  int firstElement = input[0];
      int firstOccurrence = -1;
      int secondOccurrence = -1;


      for (int i = 0; i < input.length; i++) {
          if (input[i] == firstElement) {
              if (firstOccurrence == -1) {
                  firstOccurrence = i;
              } else {
                  secondOccurrence = i;
                  break;
              }
          }
      }

      if (secondOccurrence == -1) {
          return new int[0];
      }

      int[] output = Arrays.copyOfRange(input, secondOccurrence + 1, input.length);

      return output;

 }
```

b. Use the sort, copyOf and binarySearch methods in the Arrays class to write a method that determines whether or not all of the values in a target array are also in an unsorted array. This method must not change the order of elements in the original array. To avoid changing the order when sorting, make a copy of the array first and sort and search the copy.

Example: If the array contained {1, 5, 3, 7, 2} and the targets array contained {3, 2}, the method should return true. If the targets array contained {1, 2, 3, 4} the method should return false because 4 is not in the original array.

public static boolean containsAll(int[] array, int[] targets)

```
public static boolean containsAll(int[] array, int[] targets) {

        int[] sortedArray = Arrays.copyOf(array, array.length);


        Arrays.sort(sortedArray);

        for (int target : targets) {
            int foundIndex = Arrays.binarySearch(sortedArray, target);
            if (foundIndex < 0) {

                return false;
            }
        }

        return true;
    }
```